Shell Scripting

What is Shell Script?

A shell script is a plain text file that contains series of commands & shell statements.

Types of shell:

1. Command-line shell
2. Graphical shell

```
+                                   root@
  GNU nano 6.2
For loop with seq command
for i in $(seq 0 2 10)
do
    echo "Element $i"
done
```

Types of shells

1. The Bourne Shell (sh)

   Path name: /bin/sh and /sbin/sh

- It is faster and more preferred.

- Lacks various features like:

1. Lacks ability to recall previous commands.
2. Don’t have In-built functionality to handle arithmetic and logical operation.

- Non-root user prompt is $.
- Root user prompt is #.

Types of shells

2. The C shell (csh)

Path name: /bin/csh

- This shell have numerous advantages in comparison to Bourne Shell.

- Its features are:

1. We can create alias for the commands (alias means a shortcut name).
2. We can recall previous commands.

- Non-root user prompt is $.
- Root user prompt is #.

Types of shells

3. The Korn Shell (ksh)

Path name: /bin/ksh

- Supports everything supported by Bourne Shell.

- More features:

1. Offers string, array and function manipulation similar to C.
2. In-built support for arithmetic operations.

- non-root user prompt is $.
- Root user prompt is #.

Types of shells

4. The GNU Bourne-Again Shell (bash)

Path name: /bin/bash

- It is compatible to the Bourne Shell.
- Have features from Korn and Bourne Shell.

- Features:

1. Allows us to automatically recall previously used command and edit them easily.

- Non-root user prompt is bash-VersionNumber$.
- Root user prompt is bash-VersionNumber#.

How to Check the default shell?

Command used:

echo $SHELL

```
vivek@nixcraft-asus:~$ echo "My current shell is $SHELL ($0)"
My current shell is /bin/bash (bash)
vivek@nixcraft-asus:~$
vivek@nixcraft-asus:~$ ksh
$ echo "My current shell is $SHELL ($0)"
My current shell is /bin/bash (ksh)
$ echo $SHELL
/bin/bash
$ tcsh
nixcraft-asus:~>
nixcraft-asus:~> echo $SHELL
/bin/bash
nixcraft-asus:~> echo $0
tcsh
nixcraft-asus:~> exit
exit
$ exit
vivek@nixcraft-asus:~$
```

Operators in shell scripting

1. Arithmetic Operators

- These operators are used to perform normal arithmetic operations.

- Various arithmetic operators are:

1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
5. Modulus (%)
6. Increment (++)
7. Decrement (--)

Operators in shell scripting

2. Relational Operator

- These operator define relation between two operand.

- Various relational operators are:

1. Equal to ‘==’
2. Not equal to ‘!=’
3. Less than ‘<’
4. Greater than ‘>’
5. Greater than or equal to ‘>=’

Operators in shell scripting

3. Logical Operators

- They are also known as boolean operators, and are used to perform logical operations.

- Various logical operators are:

1. Logical AND (&&)
2. Logical OR (||)
3. Not equal to (!)

Operators in shell scripting

4. Bitwise Operators

- These operators are used to perform bitwise operations.

- Various bitwise operators are:

1. Bitwise AND (&)
2. Bitwise OR (|)
3. Bitwise XOR (^)
4. Bitwise complement (-) :NOT
5. Left Shift (<<)
6. Right Shift (>>)

Operators in shell scripting

5. File Test Operator

Syntax: [-operator $file]

- These operator are used to test property of a file.

- Various file test operator are:

1. -b: It checks whether a file is block special file or not.
2. -c: It checks whether a file is character special file or not.
3. -d: It checks whether given directory exist or not.
4. -e: It checks whether given file exist or not.
5. -r: It checks whether given file has read access or not.

5. File Test Operator

6. -w: It checks whether given file has write access or not.

7. -x: It checks whether given file has execute access or not.

8. -s: It checks size of the given file. If size > 0 then output is true.

Shell Script parameter

1. Variables
2. Special Parameters

$#	Represents total argument passed to script
$0	Represents the script name
$n	Represents argument corresponding to script, positional parameter. Eg: $1, $2
$$	Represents process ID of shell in which execution is taking place
$_	Represents the command being executed previously
$?	Represents exit status of last command that was executed

Advantages of shell scripting

1. Task automation: We can automate frequently executed task.
2. To run sequence of commands as a single command.
3. Data Backup
4. Programming

How to write shell script in Linux?

1. Create a file using a file editor. Name the script with extension.sh
2. Start the script with #!/bin/sh
3. Write some code
4. Save script file as filename.sh
5. For executing the script type bash filename.sh, ./filename.sh

Functions

Functions in scripting

```
GNU nano 6.2                    function.sh *
hello ()  # definig function
 {
   echo "Hlo friends chaya pilo  $*" #'$*" indicates we can pass parameter
 }

hello #add new parameter here  #invoking function
```

- Always try to use the functions and name them properly so anyone can understand.
- Functions ensures concept of re-usability.

Format:

```
function_name()
    {
  Statements
    }
```

Deleting a function

```
root@MORALES:/home/morales/Desktop/ShellScripting# nano function.sh
root@MORALES:/home/morales/Desktop/ShellScripting# declare -f hello
hello ()
{
    echo "k cha guys chiya khani ho"
}
root@MORALES:/home/morales/Desktop/ShellScripting# unset hello
root@MORALES:/home/morales/Desktop/ShellScripting# declare -f hello
root@MORALES:/home/morales/Desktop/ShellScripting#
```

Format:

unset function_name

declare -f function name is used to see the contents of the function we defined.

Returning value from function